Feature-based Syntactic and Metric Shape Recognition

L. Prasad, A. N. Skourikhine, B. R. Schlei

Los Alamos National Laboratory, MS E541, Los Alamos, NM, 87545, USA

ABSTRACT

We present a syntactic and metric two-dimensional shape recognition scheme based on shape features.   The principal features of a shape can be extracted and semantically labeled by means of the chordal axis transform (CAT),[1, 3] with the resulting generic features, namely *torsos* and *limbs*, forming the primitive segmented features of the shape. We introduce a context-free universal language for representing all connected planar shapes in terms of their external features, based on a finite alphabet of generic shape feature primitives. Shape exteriors are then syntactically represented as strings in this language. Although this representation of shapes is not complete, in that it only describes their external features, it effectively captures shape embeddings, which are important properties of shapes for purposes of recognition. The elements of the syntactic strings are associated with attribute feature vectors that capture the metrical attributes of the corresponding features. We outline a hierarchical shape recognition scheme, wherein the syntactical representation of shapes may be "telescoped" to yield a coarser or finer description for hierarchical comparison and matching. We finally extend the syntactic representation and recognition to completely represent all planar shapes, albeit without a generative context-free grammar for this extension.

Keywords: context-free, Delaunay triangulation, feature, grammar, language, linguistic, morphology, recognition, segmentation, shape, string, syntax.

1.   INTRODUCTION

We describe a syntactic method of representing the structural outline of shapes in terms of the features that lie on their outer contours. Elsewhere in this conference, we have presented a novel geometric transform, namely the chordal axis transform (CAT), using the constrained Delaunay triangulation (CDT) of polygonized shapes.[3] This creates a paradigm for structural segmentation and analysis of shapes into morphologically meaningful components. The CAT enables the efficient parsing of shapes into semantically significant components. In particular, the CAT segments a shape into two kinds of feature primitives, namely limbs and torsos. Indeed, every planar shape may be decomposed exclusively in terms of these two kinds of basic feature primitives. The CAT of a shape, moreover, provides information about the interconnectivity of these primitives. This, then, readily yields a planar graph representation of the shape.[3]

Although the problem of planar graph isomorphism has been solved efficiently, for shape recognition purposes, it is necessary to be able to compare graphs that have significant structural similarities without being isomorphic. This more difficult problem of graph similarity is further confounded by the ill-defined notion of similarity, which can vary based on the application. Furthermore, while a planar graph encodes the topology of a shape exactly, it does not preserve information about the embedding of the shape in the plane. Indeed, a given planar graph may have more than one plane (i.e., non-self-intersecting) embedding, each corresponding to a completely different shape. For most vision-based applications, the gross features characterizing the silhouette of a shape are more important in comprehending the shape than, for example, the exact reticulation of all its features,. These considerations have motivated us to look for other more suitable characterizations of shapes in terms of their features, bearing in mind that it is important to hierarchically describe a shape in a coarse-to-fine fashion while at the same time capturing its embedding in the plane.

We outline a formal linguistic scheme for encoding shapes in terms of the feature primitives that constitute them. The feature primitives are represented symbolically and form the alphabet of the linguistic representation. A symbolic string represents the sequence of features occurring along the outer contour of a shape. Further, each character in the string is associated with an attribute vector, which encodes the corresponding feature's metrical attributes, such as length, width, area, etc. Thus, the symbolic string represents the embedding and the syntax of the shape, while the attributes capture the metrical aspects of

features. Indeed, it is possible for two distinct shapes to have identical syntactic descriptions while having very different metrical descriptions.

Attributed syntactic methods for representing and recognizing shapes have been used before.[4, 6] Typically this has involved the description of the outer boundary of shapes by means of smooth arc primitives, with attributes such as mean curvature, length, etc. Such approaches perform poorly in the presence of noise. Also, they do not directly convey or reveal the structural ramifications and hierarchies inherent in shapes. In our approach, the efficient structural segmentation of shapes into semantically salient components provided by the CAT is used to generate linguistic strings that directly reflect the structures of shapes. This yields a linguistic representation that is superior from a semantic point of view. Furthermore, these structures can be made robust to noise by subjecting them to multiscale pruning to eliminate noisy features.[3]

In representing complex patterns as strings of pattern primitives, there is a tradeoff between the complexity of the resulting language and the simplicity of the pattern primitives.[6] Although the feature primitives (limbs and torsos) obtained by the CAT are structurally elementary in that all shapes can be decomposed in terms of them, they are semantically complex enough that they generate a context-free grammar (CFG) for shape exteriors that is surprisingly simple. Moreover, the syntactic encoding of shape structure, along with the metrical attributes, permits a hierarchical parsing scheme that graduates and simplifies the recognition process. In what follows, we will describe the generative CFG that governs the feature-based language of planar shape exteriors. We will obtain a normal form for strings that eliminates rotational ambiguities. This will lead to strings of terminals, representing shapes, being recast as a hierarchical sequence of sentential forms consisting of terminals and nonterminals. This, then allows a coarse-to-fine template-matching scheme for strings, thus eliminating the need to write parsers and recognizers for each subclass of strings that correspond to different shape classes.

2. A SYNTACTIC REPRESENTATION OF SHAPE EXTERIORS

For a polygonized shape $\mathbf{P}$ (Fig.3), let CDT($\mathbf{P}$) (Fig.4) denote the set of all triangles in its constrained Delaunay triangulation.[3] The triangles of a polygon's CDT can be classified into three types, namely those with two external (i.e., polygonal boundary) edges, those with one external edge, and those with no external edges. Each kind of triangle carries morphological information about the local structure of the polygon. Accordingly, they are given different names. A triangle with two external edges marks the termination of a "limb" or a protrusion of the polygon and is called a *termination triangle* or a *T-triangle*. A triangle with one external edge constitutes the "sleeve" of a "limb" or a "torso," signifying the prolongation of the polygon, and is called a *sleeve triangle* or *S-triangle*. Finally, a triangle that has no external edges determines a junction or a branching of the polygon and will accordingly be called a *junction triangle* or a *J-triangle*. A *limb* λ is a chain complex of pairwise adjacent triangles, of the form *$TS \ldots SJ$* or *$JS \ldots ST$* (Fig.1), and a *torso* τ is a chain complex of pairwise adjacent triangles, of the form *$JS \ldots SJ$* (Fig.2). The number of sleeve triangles in a limb or a torso is allowed to be zero; thus, the duos *JT* or *TJ* also define limbs and, likewise, the duo *JJ* also defines a torso. Torsos can be further distinguished into two categories: if all the internal edges of the sequence of *S*-triangles between the two *J*-triangles of a torso connect pairs of points that belong to the same connected contour component, then the torso is termed a *stem*. Otherwise (i.e., even if one internal edge of the sequence of *S*-triangles connects a pair of points that belong to different connected contour components), it is termed a *handle*. If there are no *S*-triangles between the two *J*-triangles of a torso, then the above conditions apply for the common edge of the two *J*-triangles. Both sides of a stem can be accessed from the outside of the shape, while only one side of a handle is accessible from the outside. It is easy to see that handles occur only in shapes that have at least one hole.

The limbs, stems, and handles of a shape form its generic feature primitives. Each feature primitive is assigned a vector $\mathbf{v}$ of attributes, which may have the length, width, variance, area, etc., of the feature primitive as its components.[3] These components serve to capture the "vital-statistics" of the feature primitive (Fig.7).

We are now in a position to encode the exterior of a polygonized shape via a syntactic string of feature primitives.

Tracing (counterclockwise, say,) the outer contour of a polygonized shape, which has been decomposed into its feature primitives via its CDT, we will encounter, in sequence, the feature primitives (i.e., limbs, stems, and handles) of the shape that are adjacent to its outer contour (Fig.6). If we encounter a limb, we will record this by appending the symbol " *l* " to the string (which is initially empty); if we encounter a

handle, we will record this by appending the symbol " h " to the string. Finally, if we encounter a stem, we will record this by appending the symbol " $($ ", if this is the first time this stem has been encountered, or by the symbol " $)$ ", if this is the second time the stem has been encountered (Figs.6&7). The resulting string is a sentence in a language that characterizes the exteriors of shapes. The symbols l, h, $($, and $)$ are the terminal characters of the language. Each symbol in the string is associated with the attribute vector $v = (\lambda, \omega, \alpha)$ of the corresponding feature, where λ is the normalized length, ω is the normalized average width, and α is the normalized area of the corresponding feature primitive. Thus, we have an attributed syntactic representation of polygonized shape exteriors (Fig.7).

The above method of syntactic encoding is descriptive, in that it describes the exterior of any given shape, whose feature primitives are identified, by means of a syntactic string. One can, however, obtain a generative method of specifying shape exteriors, wherein we can generate syntactic strings that correspond to shape exteriors without the need for the *a priori* specification of a definite polygonized shape. Indeed, we will now specify a generative CFG for the language of all sentences that correspond to the elaboration of exteriors of connected shapes via feature primitives.

3. A CFG FOR FEATURE-BASED SHAPE EXTERIOR GENERATION

Starting from the syntactic description of the exterior of a simple shape, we can "grow" more complex shape exteriors by a few simple rules of morphogenesis. In essence, these rules specify the sprouting of new limbs, the metamorphosis of limbs into stems via bifurcation and handle generation, and the cloning of limbs and handles. In order to specify the structural evolution of a shape we will have to use metasymbols representing metafeatures that may evolve into composite features. Such metasymbols are called nonterminals. Intermediate stages of growth of shape exteriors will then be represented by metasentences or sentential forms, which may potentially generate infinitely many shape exteriors through the repeated application of the rules of morphogenesis. These sentential forms syntactically represent metashapes. The structural evolution of a metashape will terminate in a definite shape exterior with the replacement of all nonterminals in the corresponding sentential form by terminal symbols.

A generative CFG G for the language L_G of all possible shape exteriors is given by

$G = (\{S, L, H\}, \{l_0, h_0, l, h, (,)\}, S, R)$,

Where $\{S, L, H\}$ are the nonterminal symbols, and $\{l_0, h_0, l, h, (,)\}$ are the terminal symbols of the alphabet of G, S is the special nonterminal—the start symbol, signifying the empty sentential form, and R is the set of rules of morphogenesis.

The nonterminals L and H signify a metalimb and a metahandle, respectively, and play the role of a limb and handle in metashapes defined by sentential forms.

The set of rules of morphogenesis R is given by

R :

1. $S \rightarrow l_0$
2. $S \rightarrow h_0$ } string and loop generators

3. $L \rightarrow l$
4. $H \rightarrow h$ } terminators

5. $S \rightarrow LLL$
6. $S \rightarrow LH$ } shape initiators
7. $S \rightarrow HH$

8. $L \rightarrow LL$
9. $L \rightarrow (LL)$
10. $L \rightarrow (H)$ } limb, stem, and handle generators
11. $H \rightarrow HLH$
12. $H \rightarrow HH$

Rules 1 and 2 handle the cases when the shape has no bifurcations, but is a string-like shape (a polygon whose CDT has exactly two **T**-triangles and no **J**-triangles), or a loop with one hole (a polygon whose CDT consists of only **S**-triangles). Rules 3 and 4 terminate metalimbs and metahandles by realizing them as limbs and handles, respectively. Rules 5, 6, and 7 generate basic starting protoshapes, which may be evolved into more complex shapes by using other rules of morphogenesis. Rules 8, 9, and 10 specify how a limb may clone itself, evolve into a stem by a bifurcation, or sprout a handle through a stem, respectively. Rules 11 and 12 specify how a handle may sprout a limb onto the outside, or split into two based on an internal structural modification not seen from the outside of the shape. We will not prove here that these rules suffice to represent and generate the syntactic description of outer contours of all connected planar shapes in terms of their limb, stem, and handle feature primitives. One can define other, more restricted, *avatars* or species of these basic feature primitives, with further qualifications on their structures, in order to generate a finer description of shape exteriors. This will result in expanding the set of rules of morphogenesis, as well as the alphabet of the CFG. We will not however attempt to do so here. The above elaboration of the grammar suffices to illustrate the descriptive power and simplicity of the feature primitives identified here.

The stems are represented as a pair of parentheses, which typically enclose a substring of feature primitives that encode a component of the shape lying on one side of the stem. The stem thus distinguishes the shape into two components that are connected to each other through the stem, much like in a dumbbell. As an example, consider the string $\Sigma = C_1(C_2)C_3$, where C_1, C_2, and C_3 are substrings. The stem, denoted by the parentheses, separates the shape component described by C_2 from the shape component described by the concatenation $C_3\,C_1$. We will use this property of stems to naturally separate shape components, to perform hierarchical matching of strings. To wit, we can replace the component (C_2) by the nonterminal L and obtain the coarse resolution representation $\Sigma = C_1 L C_3$. The attribute of the nonterminal L is an aggregate of the attributes of the symbols in the substring component (C_2). Thus, we have compressed the original string Σ, by abstracting the shape component corresponding to the substring (C_2) into the nonterminal metalimb L. The resulting sentential form along with the adjusted attributes of L could then be used to compare metastrings of shapes at a coarser level of syntactic representation for a match, as if they were strings of shapes. This is the essential idea behind the hierarchical string template matching strategy we will employ in place of string recognizers and parsers. By taking advantage of the inherent structural hierarchy of shapes, we can successfully compare similar shapes with minor differences to obtain approximate matches. Indeed, two shapes may be structurally the same at a coarse level of syntactic elaboration, while differing structurally or metrically at finer levels of expression. In many application areas, such as computer vision, this facility of approximate matching is of great importance for robust recognition and understanding of objects. Before we describe this matching scheme, we must first settle some ambiguities that arise in the syntactic description of shape exteriors. Consider, again, the string $\Sigma = C_1(C_2)C_3$. An equivalent representation of this string is $\Sigma = C_2(C_3 C_1)$. This is because the outer contour of a shape is a closed curve, and as such, there is no intrinsically distinguished point on it from which we can start scanning for feature primitives. Thus, depending on where we start tracing on the outer contour, we get different representations of the same string. However, all these different representations are mere rotations of one another. In the following section, we propose a method for eliminating this ambiguity.

4. A NORMAL FORM FOR ATTRIBUTED SYNTACTIC STRINGS

A byproduct of the above ambiguity is the phenomenon observed in the case of the string Σ, where, the parentheses corresponding to the stem embrace two different components of the string in the two different representations. One way to normalize string representations would be to lexicographically order all possible rotations, and pick the first string. Indeed, giving a definite order to the alphabet of terminals induces the lexicographic ordering. This method, however, is arbitrary in that there is no natural or inherent ordering among the terminals. We prefer to use a normalization that has value from the point of shape recognition. We use the attribute information available to normalize string representations; here, we use the area attribute to do this.

We will first describe the procedure for normalizing strings, and then follow it up with the motivation behind the procedure.

Consider the substring (C_2) in the string Σ. Henceforth we will call substrings of this form, (i.e., a matching pair of parentheses, representing a stem, embracing a substring,) *compounds*. We compute the compound's cumulative area attribute by adding up all the areas of the limbs and handles in the compound, along with *half* the areas of all parentheses in the compound. This is so that we will not count areas of stems twice, since each parenthesis of a stem has all the attributes of the stem associated with it. A corresponding cumulative area, A, for the whole string can be computed in a similar manner. Now, if the cumulative area of the compound (C_2) is greater than half the cumulative area A of the string, then we "flip" the parentheses so that they now embrace the rest of the string (i.e., C_3C_1, after appropriate rotation). If, however, the compound (C_2) has cumulative area less than or equal to $A/2$, then we leave it alone–as a compound. When a compound is broken, it results in a substring that may have other compounds nested in it. Each of the *outermost* compounds of this substring is subjected to the same analysis as above. The aim of this process is to minimize the area of each outermost compound occurring in a string. Outermost compounds in any string are identified by scanning the string from left to right, and on encountering the first left parenthesis, identifying the substring enclosed by this parenthesis and its corresponding right parenthesis. This substring, along with its enclosing parentheses, forms the first outermost compound. Subsequent outermost compounds are found by continuing to scan the string and repeating this process until the end of the string is reached. It is possible that the breaking of a compound whose area is greater than $A/2$, and flipping the parentheses around the complementary substring may result in the formation of a new compound whose cumulative area is also greater than $A/2$. In this case the compound that has more elements in it is retained as a compound. It is easy to see that the above process, when applied to any string, terminates in a balanced representation that has all its outer compounds with cumulative area less than $A/2$, or, if this is not possible, its outermost compound with area greater than $A/2$ is maximized for length.

Treating each outermost compound as an indivisible entity, the string is now rotated until the element with the largest area (whether it is a compound or not) is the leftmost element in the string. The resulting string representation is our desired normal form for the string (Fig.8). As it may be clear by now, this normal form parses a shape's string into elemental and compound components such that the shape components corresponding to the outermost compounds have as even a distribution of outer area as possible. (Note that A, in general, is *not* the total area of the shape (except when the shape has no holes) but only the total area of its outer feature components.) Apart from eliminating the ambiguity in representation, reducing to normal form roughly identifies key components of the shape that need to be matched satisfactorily at a coarse level with another shape similarly represented, in order to declare similarity between them. If the strings are matched at a coarse level, then the matched compound components may be further elaborated at a finer resolution to examine for further agreement in structure. This process, thus yields a hierarchical matching procedure that is structure-driven, and allows for comparison of strings of dissimilar lengths or of slightly differing syntactic structures. Indeed, when a shape is segmented out for analysis from an image, it may not have the identical structure or syntactic expression as a stored version of the shape. But it is highly likely that their coarse-level syntactic elaborations in normal form will agree in the major components. This, then, is the motivation behind the above parsing and representation of syntactic strings of shape exteriors.

5. HIERARCHICAL MATCHING OF SHAPE EXTERIORS

After a string is cast in normal form, each of the outermost compounds is replaced by the nonterminal metalimb symbol L. The rationale behind this is that every compound may be generated from a metalimb nonterminal by repeatedly applying the morphogenesis rules 8 through 12, and finally the morphostasis rules 3 and 4. The result of this substitution is a reduced sentential form, consisting only of occurrences of the terminals l and h, and the nonterminal L. At this point, we may choose to further simplify the sentential form by making substitutions using rules 3, 4, 8, 11, and 12 in reverse. These substitutions are not applied randomly, but done in consultation with the metrical attributes of the individual symbols. For example, if a limb l occurs in the local configuration: *...hlh...*, of a reduced sentential form, and its length is insignificant in comparison with the cumulative lengths of its immediate neighbors (this may be decided based on a predetermined threshold criterion, for example,) then rule 4, in reverse, applied to its neighbors h yields the

local configuration *…HlH…*. This is then followed by the application of rule 3 in reverse to the limb *l*, resulting in the configuration *…HLH…*. We may now apply rule 12 in reverse to this trio to obtain the final configuration *…H…*. In performing these substitutions, all relevant terminals are first converted to nonterminals using rules 3 and 4. The resulting local configuration is then amenable to application of rules 8, 11, or 12, as appropriate. Each string that results from these substitution operations may be repeatedly subjected to further reduction until a satisfactory level of representational compression is achieved. Each string is processed with substitutions by fixing a resolution level determined by the threshold criterion. Subsequent processing is done at a coarser resolution, and so on. This results in a pyramid of fine-to-coarse sentential forms for the original string (Fig.8). Two strings may be compared in this scheme by first representing each string in this pyramidal form, and comparing corresponding levels of coarse representation in increasing order of detail. The strings are considered for further matching only if their representations at the coarsest scale are the same syntactically, and their corresponding attributes are within a prespecified tolerance of each other. Terminal symbols that are matched in the two strings at any level are not considered again while matching at the next level. However, if two nonterminals of the two strings are matched, then the corresponding elaborations of these nonterminals are matched for similarity at the next level. In this way, we minimize the number of comparisons necessary for hierarchical matching. When two metalimb nonterminals, corresponding to compounds at the next level, are matched, then at the subsequent level the two compounds are matched as if they are two new strings; i.e, they have their own pyramids. Indeed, if (C_1) and (C_2) are the two compounds in the two strings, then these are treated as two new strings given by lC_1 and lC_2. Here, terminal limbs at the beginning of the enclosed strings replace the stems of the two compounds. The limbs inherit the attributes of the corresponding stems. This syntactic operation is equivalent to the geometrical operation of severing the shape components represented by the compounds from the parent shapes. The stems then become limbs in the separated shape components, which are now shapes in their own right. Pyramids are constructed for these shapes and matched hierarchically as described before.

We have indicated how string simplification and hierarchical template matching may be done, without putting too fine a point on it; the user and the application at hand best dictate the exact implementation of this scheme. This hierarchical matching scheme allows for similar by slightly differing (in syntax, length, and attributes) strings to be matched robustly, and even understand at what level of resolution and what spatial locations the strings differ. This has significant value to the problem of image understanding. Elsewhere, in a later work, we will describe the construction of an attributed syntactic shape database, which serves as a hierarchical repository of learned shapes. Such a database, in conjunction with an implementation of the above-described matching procedure, serves to recognize shapes in imagery from vision sensors and various offline sources as well. An interesting possible application would be the design of a smart search engine for image data.

6.   VARIATIONS AND EXTENSIONS OF SYNTACTIC SHAPE REPRESENTATION

In many applications, the syntactic characterization of the exterior of a shape, without regard to holes, may be sufficient. In this case, the CDT is performed only for the outer contour, with the shape being treated as if it were simply connected. Here, either the same grammar may be used, or the grammar may be further simplified by eliminating handles and metahandles and we can generate a CFG with a smaller set of production rules.

A complete description of a shape with holes is also possible with the method of syntactic encoding of shape exteriors that we have introduced. However, in this case, the CFG *G* cannot *generate* all possible shapes with holes completely. Indeed, it has been shown[5] that a CFG cannot generate all planar graphs (every planar shape induces a planar graph via its skeleton (Fig.5)). Nevertheless, since we do not depend on the grammar's context-freeness to perform recognition, we will be content with a complete syntactic representation of shapes. The recognition procedure described above also extends to this complete representation case without significant modifications. In essence, this complete representation is obtained by encoding each inner hole contour via the feature primitives occurring along it. The shape may then be syntactically represented by a set of strings, each for one hole, and one string for the outer contour. For an unambiguous representation, however, we have to introduce an ordering on these strings, so that two identical shapes have their strings listed in the same order. This will ensure that an exact matching indeed corresponds to the case when the two shapes are identical. To achieve this ordering, we will obtain a

canonical numbering of the holes of a shape, based on the normal form of its outer contour's attributed syntactic string. The outer contour will be given the number 0. If the shape has $n>0$ holes, then we first number these holes randomly, using distinct numbers greater than n. Each point of the shape's boundary inherits the number of the contour on which it occurs. In the CDT of the shape, each identified limb or stem feature primitive of the shape will acquire a contour number (as an added attribute) based on the contour on which its sleeve triangle edges lie. In the case of handles, there will be two contour numbers associated since a handle's sleeve triangles straddle exactly two contours. As before, when the feature primitive has no sleeve triangles, the common edge between the two *J*-triangles (for stems and handles) or the *J*-triangle and the *T*-triangle (for limbs) is used to determine these contour numbers. We now consider the full syntactic string of the shape's outer contour in normal form, and without substitutions. This string is scanned from left to right, until the first handle is encountered. This handle will have two contour numbers associated with it, namely 0 and a number m greater than n. We change this and every occurrence of m to 1, in all the attribute vectors of all the feature primitives of the shape. In effect, we will have renumbered contour m as contour 1. The string corresponding to contour 1 is now written, starting from the handle that was used to obtain its new number (to ensure normalization of this string's representation), and moving along the contour in a clockwise direction. The outer contour's string is scanned further for the next occurrence of a handle, one of whose contour numbers is greater than n. This contour number, and all occurrences of it will be given the number 2. Subsequently, the string corresponding to contour 2 will be written, as before. This process is continued until every handle has contour numbers less than or equal to n in the outer contour's syntactic string. If, at the end of this process, we have renumbered all the hole contours, then we will have the canonical numbering of holes, along with their syntactic strings occurring in the numbered order. If not, we continue to scan in the string corresponding to contour 1 for the next available handle with a contour number greater than n and assign all occurrences of this number the least unused number less than n. Thus, after we exhaustively renumber the holes and produce their strings in order, we have an unambiguous, complete attributed syntactic representation of a shape in terms of $n+1$ ordered strings. We may choose to represent the shape by a single composite string by concatenating the $n+1$ strings, separated pairwise by a special symbol. The hierarchical matching algorithm described before can easily be adapted for this complete representation also.

7. CONCLUSIONS

We have presented an attributed syntactic representation and a hierarchical recognition scheme for planar shapes based on their morphological structure. There are several improvements and modifications possible in this basic scheme. The flexibility of this scheme serves a wide range of application areas. These include computer vision, pattern analysis, artificial intelligence, document analysis, optical handwritten character recognition, biometrics, robotic navigation, automated surveillance, and content-based image data retrieval. This work is part of an ongoing effort to automate the intelligent assessment of image data for surveillance and remote sensing applications.

8. ACKNOWLEDGEMENTS

REFERENCES

1.   L. Prasad, "Morphological Analysis of Shapes," CNLS Newsletter, No. 139, July 1997, LALP-97-010-139, Center for Nonlinear Studies, Los Alamos National Laboratory.
2.   L. Prasad, R. L. Rao "Multi-scale Discretization of Shape Contours," in Mathematical Imaging, **4117** Vision Geometry IX, *Proc. of the 45th SPIE Annual Meeting*, San Diego, CA, 2000.
3.   L. Prasad, R. L. Rao "A Geometric Transform for Shape Feature Extraction," in Mathematical Imaging, **4117** Vision Geometry IX, *Proc. of the 45th SPIE Annual Meeting*, San Diego, CA, 2000.
4.   K. C. You, K. S. Fu "A Syntactic Approach to Shape Recognition Using Attributed Grammars," IEEE Trans. Systems, Man and Cybernetics, Vol. SMC-**9**, No. **6**, 334-345 (June 1979).
5.   T. Pavlidis "Linear and Context-Free Grammars" J .ACM, Vol. 19, No. 1, pp. 11−22 (Jan. 1972)
6.   K. S. Fu "Linguistic Approach to Pattern Recognition," in *Applied Computation Theory: Analysis, Design, Modeling* Ed. R. T. Yeh, Prentice Hall (1976).
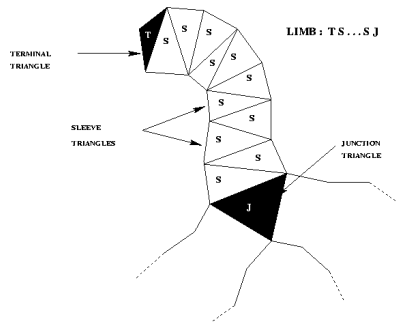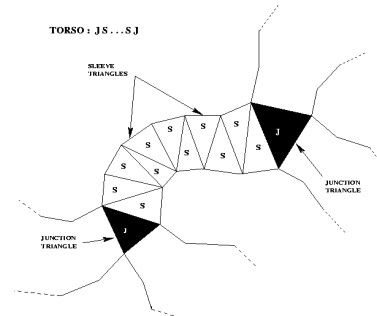
Fig.1 A limb chain complex.
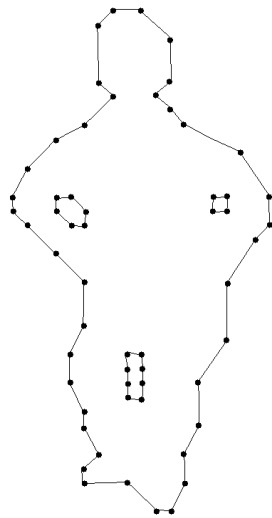

Fig.2 A torso chain complex.
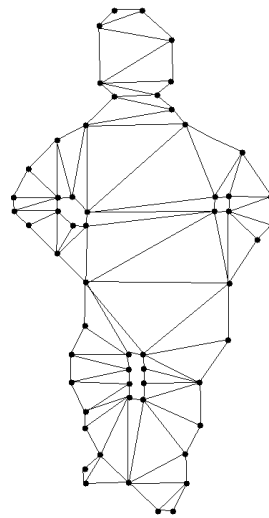

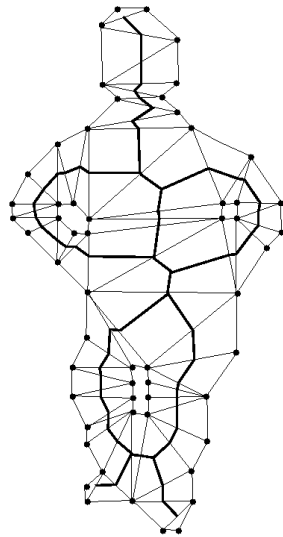Fig.3 Shape with three holes.

Fig.4 CDT of the shape.

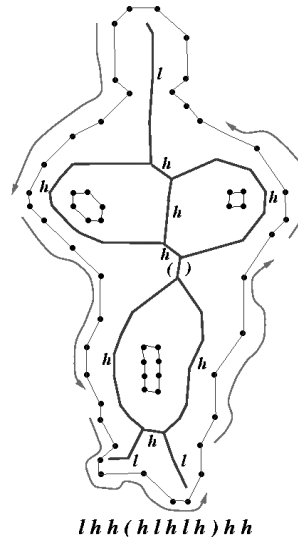Fig.5 Feature primitives highlighted by
the shape's skeleton.

l h h (h l h l h) h h

Fig.6 Syntactic feature labeling and
string representation of shape exterior.

The Grammar G :

$G = (\ \{S, L, H\}, \{l_0, h_0, l, h, (,) \}, S, R\ )$

The Production Rule Set R :

$S \longrightarrow l_0$ **String & Loop**
$S \longrightarrow h_0$ **Generators**

$L \longrightarrow l$ **Terminators**
$H \longrightarrow h$

$S \longrightarrow LLL$
$S \longrightarrow LH$ **Shape Initiators**
$S \longrightarrow HH$

$L \longrightarrow LL$
$L \longrightarrow (LL)$ **Stem, Limb & Handle**
$L \longrightarrow (H)$ **Generators**

$H \longrightarrow HH$
$H \longrightarrow HLH$

Human Shape

$l\,l\,(\,(\,l\,l\,)\,l\,)$

Feature	α	λ	ω
l	202.312	20.524	9.857
l	286.204	41.111	6.962
$($	49.714	4.590	21.664
$($	165.397	20.480	16.152
l	366.757	50.986	7.193
l	333.293	45.503	7.325
$)$	165.397	20.480	16.152
l	274.325	39.921	6.872
$)$	49.714	4.590	21.664

α = Area of the feature primitive

λ = Length of the feature primitive

ω = Mean width of the feature primitive

Fig.7 Syntactic representation of a human shape and the corresponding feature attributes.

Trunk and feet	Left wing	Head and neck	Right wing
27	13	15	13
L	*L*	*L*	*L*

Coarse ↑ *Representation*

Trunk and feet	Left wing	Head and neck	Right wing
27	13	15	13
((l l l) (l l l))	(l l)	(h h l h h l h h)	(l l)

Normal ↑ *Form*

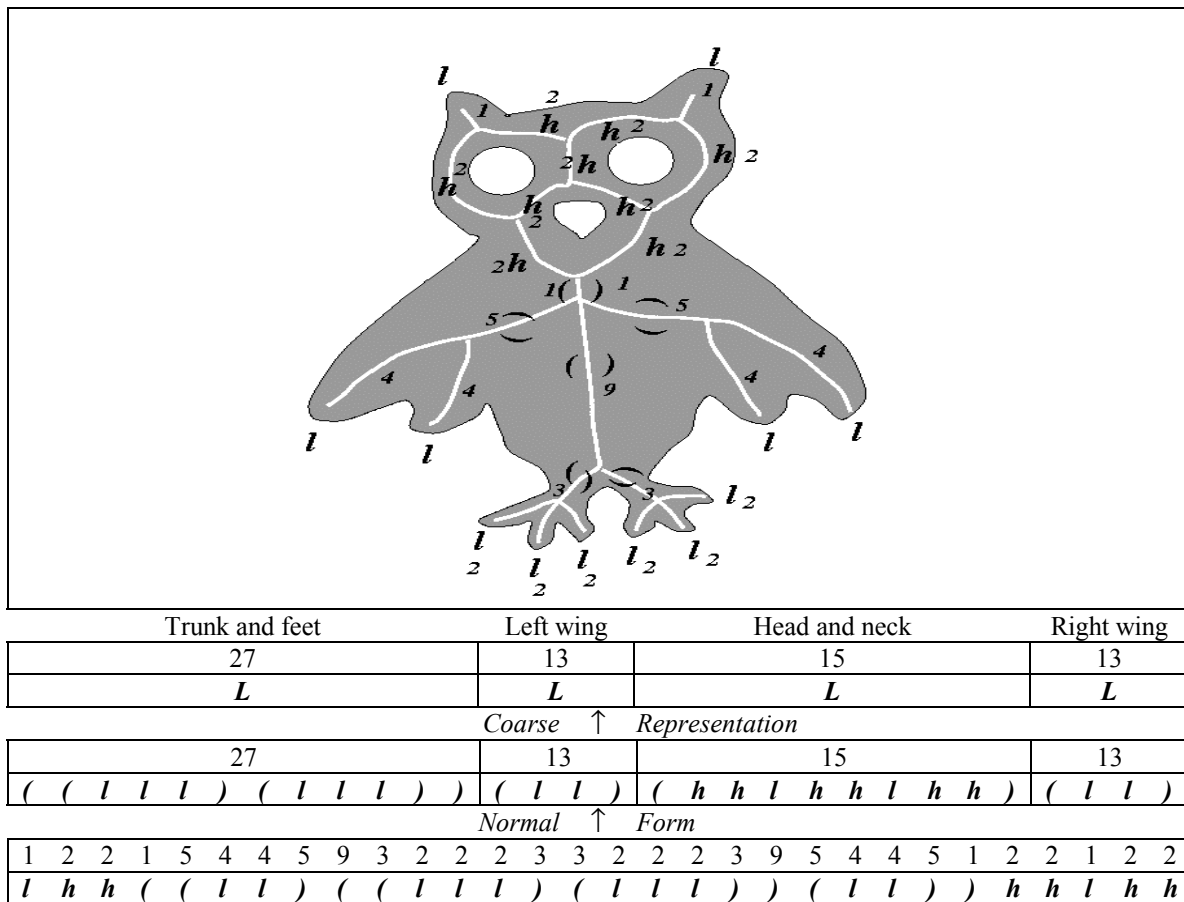| 1	2	2	1	5	4	4	5	9	3	2	2	2	3	3	2	2	2	3	9	5	4	4	5	1	2	2	1	2	2
l	h	h	(	(	l	l	)	(	(	l	l	l	)	(	l	l	l	)	)	(	l	l	)	)	h	h	l	h	h

Fig.8 Hierarchical fine-to-coarse pyramidal representation of the attributed syntactic string of a shape.

9